

Resolving the Complexity of Some Data Privacy Problems

Jeremiah Blocki¹ and Ryan Williams²

¹ Carnegie Mellon University jblocki@andrew.cmu.edu

² IBM Almaden Research Center rrwilliams@gmail.com

*An extended abstract of this work will appear in ICALP 2010.

Abstract. We formally study two methods for data sanitation that have been used extensively in the database community: k -anonymity and ℓ -diversity. We settle several open problems concerning the difficulty of applying these methods optimally, proving both positive and negative results:

- 2-anonymity is in P.
- The problem of partitioning the edges of a triangle-free graph into 4-stars (degree-three vertices) is NP-hard. This yields an alternative proof that 3-anonymity is NP-hard even when the database attributes are all *binary*.
- 3-anonymity with only 27 attributes per record is MAX SNP-hard.
- For databases with n rows, k -anonymity is in $O(4^n \cdot \text{poly}(n))$ time for all $k > 1$.
- For databases with ℓ attributes, alphabet size c , and n rows, k -Anonymity can be solved in $2^{O(k^2(2c)^\ell)} + O(n\ell)$ time.
- 3-diversity with binary attributes is NP-hard, with one sensitive attribute.
- 2-diversity with binary attributes is NP-hard, with three sensitive attributes.

1 Introduction

The topic of *data sanitization* has received enormous attention in recent years. The high-level idea is to release a database to the public in such a manner that two conflicting goals are achieved: (1) the data is useful to benign researchers who want to study trends and identify patterns in the data, and (2) the data is not useful to malicious parties who wish to compromise the privacy of individuals. Many different models for data sanitization have been proposed in the literature, and they can be roughly divided into two kinds: *output perturbative* models (e.g., [1, 2]) and *output abstraction* models (e.g., [3–5]). In perturbative models, some or all of the output data is perturbed in a way that no longer corresponds precisely to the input data (the perturbation is typically taken to be a random variable with nice properties). This includes work which assumes *interaction* between the prospective data collector and the database, such as differential privacy. In abstraction models, some of the original data is suppressed or generalized (e.g. an age becomes an age range) in a way that preserves data integrity. The latter models are preferred in cases where data integrity is the highest priority, or when the data is simply non-numerical.

In this work, we formally study two data abstraction models from the literature, and determine which cases of the problems are efficiently solvable. We study k -anonymity and ℓ -diversity.

1.1 K-Anonymity

The method of k -anonymization, introduced in [3, 4], is a popular method in the database community for publicly releasing part of a database while protecting individual identities in that database. Formally speaking, an instance of the k -anonymity problem is a matrix (a.k.a. database) with n rows and m columns with entries drawn from an underlying alphabet. Intuitively, the rows correspond to individuals and the columns correspond to various attributes of them. For hardness results, we study a special case called the *suppression model*, where the goal is to replace entries in the matrix with a special symbol \star (called a ‘star’), until each row is identical to at least $k - 1$ other rows. The intuition is that the information released does not explicitly identify any individual in the database, but rather identifies at worst a group of size k .³

³ This intuition can break down when combined with background knowledge [5]. However, our intent in this paper is not to critique the security/insecurity of these methods, but rather to understand their feasibility.

A trivial way to k -anonymize a database is to suppress every entry (replacing all entries with \star), but this renders the database useless. In order to maximize the utility of the database, one would like to suppress the fewest entries—this is the k -ANONYMITY problem with suppression. Meyerson and Williams [6] proved that in the most general case, this is a difficult task: k -ANONYMITY is NP-hard for $k \geq 3$, provided that the size of the alphabet is $\Omega(n)$. Aggarwal *et al.* [7] improved this, showing that 3-ANONYMITY remains NP-hard even when the alphabet size is 3. Bonizzoni *et al.* [8] further improved the result to show that 3-ANONYMITY is APX-hard, even with a binary alphabet. They also showed that 4-ANONYMITY with a constant number of attributes per record is NP-hard. Two basic questions remain:

1. How difficult is 3-anonymity with a small number of attributes per record?
2. How difficult is the 2-anonymity problem?

Addressing the two questions above, we discover both a positive and negative result. On the positive side, in Section 3 we present a polynomial time algorithm for 2-ANONYMITY, applying a result of Anshelevich and Karagiozova [9]:

Theorem 1. 2-ANONYMITY is in P.

The polynomial time algorithm works not only for the simple suppression model, but also for the most general version of k -anonymity, where for each attribute we are given a *generalization hierarchy* of possible ways to withhold data.

In Section 4, we consider k -anonymity in databases where the number of attributes per record is constant. This setting seems to be the most relevant for practice: in a database of users, the number of attributes per user is often dwarfed by the number of users in the database. We find a surprisingly strong negative result.

Theorem 2. 3-ANONYMITY with just 27 attributes per record is MAX SNP-hard. Therefore, 3-ANONYMITY does not have a polynomial time approximation scheme in this case, unless P = NP.

The proof uses an alphabet with $\Omega(n)$ cardinality. This motivates the question: how efficiently can we solve k -anonymity with a small alphabet and constant number of attributes per record? Here we can prove a positive result, showing that when the number of attributes is small and the alphabet is constant, there are subexponential algorithms for optimal k -anonymity for every $k > 1$.

Theorem 3. For every $k > 1$, an optimal k -anonymity solution can be computed in $O(4^n \text{poly}(n))$ time, where n is the total number of rows in the database.

Theorem 4. Let ℓ be the number of attributes in a database, let c be the size of its alphabet, and let n be the number of rows. Then k -Anonymity can be solved in $2^{O(k^2(2c)^\ell)} + O(n\ell)$ time.

This improves on results in [10]. Theorem 4 implies that k -Anonymity is solvable in polynomial time whenever $\ell \leq (\log \log n)/\log c$ and $c \leq \log n$. Theorem 4 also implies that for $c = n^{o(1)}$ and $\ell = O(1)$, k -anonymity is solvable in *subexponential time*. Therefore it is highly unlikely that we can tighten the unbounded alphabet constraint of Theorem 2, for otherwise all of NP has $2^{n^{o(1)}}$ time algorithms.

In Section 5, we provide an alternative proof that BINARY 3-ANONYMITY, the special case of the problem where all of the attributes are binary-valued, is NP-hard. This result is weaker than [8] who recently showed that BINARY 3-ANONYMITY is APX-hard. However, our proof also shows that a certain edge partitioning problem is NP-complete, which to the best of our knowledge is new⁴. Let EDGE PARTITION INTO TRIANGLES AND 4-STARS be the problem of partitioning the edges of a given graph into 3-cliques (triangles) and 4-stars (graphs with three degree-1 nodes and one degree-3 node).

Theorem 5. EDGE PARTITION INTO TRIANGLES AND 4-STARS is NP-complete.

Theorem 5 implies that the TERNARY 3-ANONYMITY hardness reduction given in [7] is sufficient to conclude that BINARY 3-ANONYMITY is NP-hard.

⁴ EDGE PARTITION INTO TRIANGLES is NP-Complete as is EDGE PARTITION INTO 4-STARS [11], but this does not imply that EDGE PARTITION INTO TRIANGLES AND 4-STARS is NP-Complete.

1.2 L-Diversity

Finally, in Section 6 we consider the method of ℓ -diversity introduced in [5], which has also been well-studied. This method attempts to refine the notion of k -anonymity to protect against knowledge attacks on particular sensitive attributes.

We will work with a simplified definition of ℓ -diversity that captures the essentials. Similar to k -anonymity, we think of an ℓ -diversity instance as a table (database) with m rows (records) and n columns (attributes). However, each attribute is also given a label q or s , inducing a partition of the attributes into two sets Q and S . Q is called the set of quasi-identifier attributes and S called the set of sensitive attributes.

Definition 1. *A database D is said to be ℓ -diverse if for every row u_0 of D there are (at least) $\ell - 1$ distinct rows $u_1, \dots, u_{\ell-1}$ of D such that:*

1. $\forall q \in Q, 0 \leq i < j < \ell$ we have $u_i[s] = u_j[s]$
2. $\forall s \in S, 0 \leq i < j < \ell$ we have $u_i[s] \neq u_j[s]$

Constraint 1 is essentially the same as k -anonymity. Any row must have at least $k - 1$ other rows whose (non sensitive) attributes are identical. Intuitively, Constraint 2 prevents anyone from definitively learning any row's sensitive attribute; in the worst case, an individual's attribute can be narrowed down to a set of at least ℓ choices. Similar to k -anonymity with suppression, we allow stars to be introduced to achieve the two constraints.

We can show rather strong hardness results for ℓ -diversity.

Theorem 6. *Optimal 2-diversity with binary attributes and three sensitive attributes is NP-hard.*

Theorem 7. *Optimal 3-diversity with binary attributes and one sensitive attribute is NP-hard.*

Independent of their applications in databases, k -anonymity and ℓ -diversity are also interesting from a theoretical viewpoint. They are natural combinatorial problems with a somewhat different character from other standard NP-hard problems. They are a kind of discrete partition task that has not been studied much: *find a partition where each part is intended to "blend in the crowd."* Such problems will only become more relevant in the future, and we believe the generic techniques developed in this paper should be useful in further analyzing these new partition problems.

2 Preliminaries

We use $\text{poly}(n)$ to denote a quantity that is polynomial in n .

Definition 2. *Let n and m be positive integers. Let Σ be a finite set. A database with n rows (records) and m columns (attributes) is a matrix from $\Sigma^{n \times m}$. The alphabet of the database is Σ .*

Definition 3. *Let k be a positive integer. A database is said to be k -anonymous or k -anonymized if for every row r_i there exist at least $k - 1$ identical rows.*

As mentioned earlier, there are two methods of achieving k -anonymity: suppression and generalization. In the suppression model, cells from the table are replaced with stars until the database is k -anonymous. Informally, the generalization model allows the entry of an individual cell to be replaced by a broader category. For example, one may change a numerical entry to a range, *e.g.* (Age: 26 \rightarrow Age: [20-30]). A formal definition is given in Section 3.1.

In our hardness results, we consider k -anonymity with suppression. Since suppression is a special case of generalization, the hardness results also apply to k -anonymity with generalization. Interestingly, our polynomial time 2-anonymity algorithm works under both models.

Definition 4. *Under the suppression model, the cost of a k -anonymous solution to a database is the number of stars introduced.*

We let $\text{Cost}_k^*(D)$ denote the minimum cost of k -anonymizing database D .

For the proofs of hardness, we introduce a few graph theoretical notions. Recall K_n denotes the complete graph on n vertices; we use the word *triangle* to denote K_3 .

Definition 5. Let $k \geq 1$. A k -star is a simple graph with $k - 1$ edges, all of which are incident to a common vertex v . v is called the center of the k -star. The other $k - 1$ vertices are called the leaves of the k -star.

We also need a particular type of graph which we call a 3-binary tree. All interior nodes of such a tree have degree three.

Definition 6. Let $d \in \mathbb{N}$ be given. A 3-binary tree of depth d is a complete tree of depth d where the root has three children and all other nodes have two children.

For our inapproximability results, we need the notion of an L-reduction [12].

Definition 7. Let A and B be two optimization problems and let $f : A \rightarrow B$ be a polynomial time computable transformation. f is an L-reduction if there are positive constants α and β such that

1. $\text{OPT}(f(I)) \leq \alpha \cdot \text{OPT}(I)$
2. For every solution of $f(I)$ of cost c_2 we can in polynomial time find a solution of I with cost c_1 such that

$$|\text{OPT}(I) - c_1| \leq \beta \cdot |\text{OPT}(f(I)) - c_2|$$

3 Polynomial Time Algorithm for 2-Anonymity

Because 3-anonymity is hard even for binary attributes, it is natural to wonder if 2-anonymity is also difficult. However it turns out that achieving optimal 2-anonymity is polynomial time solvable. The resulting algorithm is nontrivial and would require heavy machinery to implement. We rely on a special case of hypergraph matching called **SIMPLEX MATCHING**, introduced in [9].

Definition 8. **SIMPLEX MATCHING:** Given a hypergraph $H = (V, E)$ with hyperedges of size 2 and 3 and a cost function $c : E \rightarrow \mathbb{N}$ such that

1. $(u, v, w) \in E(H) \implies (u, v), (v, w), (u, w) \in E(H)$ and
2. $c(u, v) + c(v, w) + c(u, w) \leq 2 \cdot c(u, v, w)$

find $M \subseteq E$ such that for all $v \in V$ there is a unique $e \in M$ containing v , and $\sum_{e \in M} c(e)$ is minimized.

Anshelevich and Karagiozova gave a polynomial time algorithm to solve **SIMPLEX MATCHING**. We show that 2-ANONYMITY can be efficiently reduced to a simplex matching.

Reminder of Theorem 1. 2-ANONYMITY is in P.

Proof. Given a database D with rows r_1, \dots, r_n , let $C_{i,j}$ denote the number of stars needed to make rows r_i and r_j . Similarly define $C_{i,j,k}$ to be the number of stars needed to make r_i, r_j, r_k all identical. Observe that in a 2-anonymization, any group with more than three identical rows could simply be split into subgroups of size two or three without increasing the anonymization cost. Therefore we may assume (without loss of generality) that the optimal 2-anonymity solution partitions the rows into groups of size two or three.

Construct a hypergraph H as follows:

1. For every row r_i of D , add a vertex v_i .
2. For every pair r_i, r_j , add the 2-edge $\{v_i, v_j\}$ with cost $c(v_i, v_j) = C_{i,j}$.
3. For every triple r_i, r_j, r_k , add the 3-edge $\{v_i, v_j, v_k\}$ with cost $c(v_i, v_j, v_k) = C_{i,j,k}$.

Thus H is a hypergraph with n vertices and $O(n^3)$ edges. We claim that H meets the conditions of the simplex matching problem. The first condition is trivially met. Suppose we anonymize the pair of rows r_i, r_j with cost $C_{i,j}$, so both rows have $\frac{1}{2}C_{i,j}$ stars when anonymized. Observe that if we decided to anonymize the group r_i, r_j, r_k , the number of stars introduced per row would not decrease. That is, for all i, j, k we have

$$\frac{1}{3} \cdot C_{i,j,k} \geq \frac{1}{2} \cdot C_{i,j}.$$

By symmetry, we also have

$$\frac{1}{3} \cdot C_{i,j,k} \geq \frac{1}{2} \cdot C_{j,k}, \quad \frac{1}{3} \cdot C_{i,j,k} \geq \frac{1}{2} \cdot C_{i,k}.$$

Adding the three inequalities together,

$$C_{i,j,k} \geq \frac{1}{2}(C_{i,j} + C_{j,k} + C_{i,k}).$$

Therefore H is an instance of the simplex matching problem.

Finally, observe that any simplex matching of H corresponds to a 2-anonymization of D with the same cost, and vice-versa. \square

3.1 The general case

The proof of Theorem 1 also carries over to the most general case of 2-ANONYMITY, where instead of only suppressing entries with stars, we have a *generalization hierarchy* of possible values to write to an entry. We give a simple definition of generalization hierarchy that captures the essential features described in [4].

Definition 9. Let Σ be an alphabet of attributes, and let $\Gamma \supseteq \Sigma$. A generalization hierarchy is a rooted tree T on $|\Gamma|$ nodes with $|\Sigma|$ leaves, where the vertices v in T are put in one-to-one correspondence with alphabet symbols $a(v) \in \Gamma$, the leaves are in one-to-one correspondence with the symbols of Σ , and all vertices v have a cost $c(v) \in \mathbb{N}$. The cost function satisfies the property that if u is the parent of v in T , then $c(u) \geq c(v)$.

The key property of a generalization hierarchy is that the cost function decreases as one moves from the root of T down to its leaves. Note the suppression model of k -ANONYMITY can be modeled with a trivial generalization hierarchy: we can take a star graph T where the center of the star has symbol \star and cost 1, while the leaves (corresponding to the letters of Σ) have cost 0. For any generalization hierarchy T , one can define the k -ANONYMITY- T problem, where the goal is to replace some entries in a matrix from $\Sigma^{n \times m}$ with symbols in $\Gamma - \Sigma$ such that (a) every row is identical to at least $k - 1$ other rows, (b) every symbol replaced is a *successor* of the new alphabet symbol replacing it (in T), and (c) the total sum of costs associated with these new symbols is minimized.

Theorem 8. For every generalization hierarchy T , k -ANONYMITY- T is in P.

Proof. (Sketch) We define a hypergraph H just as in Theorem 1, but with new costs $C_{i,j}$ and $C_{i,j,k}$ reflecting the costs of a particular generalization hierarchy. One can still prove that the conditions for the simplex matching problem hold, using the fact that if u is any ancestor of v , then $c(u) \geq c(v)$. This condition implies that if we have anonymized two rows r_i, r_j , adding a third row r_k to be anonymized cannot *decrease* the cost of anonymization per row. That is, the particular generalization symbols needed to make all three rows identical could only cost more than the symbols needed to anonymize the two rows originally. \square

4 k -Anonymity With Few Attributes

We now turn to studying the complexity of k -Anonymity with a constant number of attributes. First we show that for an unbounded alphabet, the 3-anonymity problem is still hard even with only 27 attributes. We use the following MAX SNP-hard problem in our proof.

Definition 10. MAX 3DM-3 (Maximum 3-Dimensional Matching With 3 Occurrences)

INSTANCE: A set $M \subseteq W \times X \times Y$ of ordered triples where W, X and Y are disjoint sets. The number of occurrences in M of any element in W, X or Y is bounded by 3. Let

$$C_{3DM}(M') = \frac{3|M'|}{|W| + |X| + |Y|}.$$

GOAL: Maximize $C_{3DM}(M')$ over all $M' \subseteq M$ such that no two elements of M' agree in any coordinate.

Reminder of Theorem 2. 3-ANONYMITY with just 27 attributes per record is MAX SNP-hard. Therefore, 3-ANONYMITY does not have a polynomial time approximation scheme in this case, unless $P = NP$.

Proof. To show 3-anonymity is MAX SNP-hard, we show that there is an L-reduction from MAX 3DM-3 to 3-ANONYMITY WITH 27 ATTRIBUTES [12], since it is known that MAX 3DM-3 is MAX SNP-complete [13].

Given a MAX 3DM-3 instance $I = (M, W, X, Y)$, construct a 3-ANONYMITY instance D as follows:

1. Define $\Sigma = M \cup W \cup X \cup Y$, so that it contains a special symbol for each triple in $t \in M$ and each element $r \in W \cup X \cup Y$.
2. Add a row to D corresponding to each element $r_i \in W \cup X \cup Y$, as follows. For $r \in W \cup X \cup Y$, let $t_{r,1}, t_{r,2}, t_{r,3} \in M$ be the three triples of M which contain r (if there are less than three triples then simply introduce new symbols).

– If $r \in W$ then add the following row to D :

$$\boxed{t_{r,1} \mid t_{r,1} \mid t_{r,1} \mid t_{r,1} \mid t_{r,1} \mid t_{r,1} \mid t_{r,1} \mid t_{r,1} \mid t_{r,1} \mid t_{r,2} \mid t_{r,2} \mid \dots \mid t_{r,3}}$$

That is, the row contains nine copies of $t_{r,1}$, nine copies of $t_{r,2}$, then nine copies of $t_{r,3}$.

– If $r \in X$, then add the row:

$$\boxed{t_{r,1} \mid t_{r,1} \mid t_{r,1} \mid t_{r,2} \mid t_{r,2} \mid t_{r,2} \mid t_{r,3} \mid t_{r,3} \mid t_{r,3} \mid t_{r,1} \mid t_{r,1} \mid \dots \mid t_{r,3}}$$

– If $r \in Y$, then add the row:

$$\boxed{t_{r,1} \mid t_{r,2} \mid t_{r,3} \mid t_{r,1} \mid t_{r,2} \mid t_{r,3} \mid t_{r,1} \mid t_{r,2} \mid t_{r,3} \mid t_{r,1} \mid t_{r,2} \mid \dots \mid t_{r,3}}$$

Suppose $w_i \in W, x_j \in X, y_k \in Y$ are arbitrary. Then the corresponding three rows in the database have the form:

$$\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|} \hline w_i & t_{w_i,1} & t_{w_i,1} & t_{w_i,1} & t_{w_i,1} & t_{w_i,1} & t_{w_i,1} & t_{w_i,1} & t_{w_i,1} & t_{w_i,1} & t_{w_i,1} & \dots & t_{w_i,3} \\ \hline x_j & t_{x_j,1} & t_{x_j,1} & t_{x_j,1} & t_{x_j,2} & t_{x_j,2} & t_{x_j,2} & t_{x_j,2} & t_{x_j,3} & t_{x_j,3} & t_{x_j,3} & \dots & t_{x_j,3} \\ \hline y_k & t_{y_k,1} & t_{y_k,2} & t_{y_k,3} & t_{y_k,1} & t_{y_k,2} & t_{y_k,3} & t_{y_k,1} & t_{y_k,2} & t_{y_k,3} & t_{y_k,1} & \dots & t_{y_k,3} \\ \hline \end{array}$$

Observe that D has a total of $27n$ entries, where $n = |X| + |W| + |Y|$. Recall $Cost_3^*(D)$ is the optimal number of stars needed to 3-anonymize D . It is useful to redefine 3-Anonymity as maximization problem (where one maximizes the information released). Let P be a 3-anonymous solution to D , and define

$$C_{3ANON}(P) = 1 - \frac{Cost_3^*(P)}{27n},$$

so that $OPT(D) = \max_P \{C_{3ANON}(P)\}$.

Suppose $D = \{r_1, \dots, r_n\}$ is an instance of 3-ANONYMITY obtained from the above reduction. Three properties are immediate from the construction of D :

1. For any x rows r_i, r_j, r_k, r_l , where $x \geq 4$, the cost of anonymizing these rows is

$$C_{i,j,k,l} = 27x$$

because there is no alphabet symbol that is used in all 4 rows.

2. If $\{r_i, r_j, r_k\} \notin M$ then the cost of anonymizing the three corresponding rows is

$$C_{i,j,k} = 3 \cdot 27 = 81$$

because there is no alphabet symbol that is used in all 3 rows.

3. If $\{r_i, r_j, r_k\} \in M$ then the cost of anonymizing the three corresponding rows is

$$C_{i,j,k} = 3 \cdot 26 = 78$$

because the three rows match in exactly one of the 27 columns.

These properties lead directly to the lemma:

Lemma 1. *There is a polynomial time mapping g from 3DM-3 feasible solutions to 3-anonymity feasible solutions, such that if $M' \subseteq M$ is a 3DM-3 feasible solution then $C_{3DM}(M') = 27C_{3ANON}(g(M'))$.*

The proof of Lemma 1 is given in Appendix A.

It remains for us to show that the above reduction is in fact an L -reduction. Let I be a MAX 3DM-3 instance, with corresponding 3-Anonymity instance $f(I)$, and set $\alpha = \frac{1}{27}, \beta = 27$. Now by Lemma 1

$$OPT(f(I)) = \frac{1}{27}OPT(I) \leq \alpha OPT(I)$$

so that condition (1) of an L -reduction holds. Similarly, if we have a solution of $f(I)$ of cost c_2 , then again by Lemma 1 we can quickly compute a solution to I of cost $c_1 = 27c_2$. Therefore,

$$|OPT(I) - c_1| = |27OPT(f(I)) - 27c_2| = \beta |OPT(f(I)) - c_2|$$

so that condition (2) also holds. □

To complement the above bad news, we now give an efficient algorithm for optimal k -anonymity when the number of attributes and the size of the alphabet are both small. Along the way, we also give an algorithm for the general k -anonymity problem that runs in roughly 4^n time (again n is the number of rows).

A naive algorithm for k -anonymity would take an exorbitant amount of time, trying all possible partitions of n rows into groups with cardinality between k and $2k - 1$. We can reduce this greatly using a divide-and-conquer recursion.

Reminder of Theorem 3. *For every $k > 1$, an optimal k -anonymity solution can be computed in $O(4^n \text{poly}(n))$ time, where n is the total number of rows in the database.*

Proof. Interpret our k -anonymity instance S as a multiset of n vectors drawn from $|\Sigma|^\ell$. Define $S_k = \{T : T \subseteq S, |T| \in [\frac{n}{2}, \frac{n}{2} + 2k]\}$. That is, S_k contains all multisubsets which have approximately $n/2$ elements. Then

$$Cost_k(S) = \operatorname{argmin}_{T \in S_k} [Cost_k(S - T) + Cost_k(T)], \quad (1)$$

where $Cost_k(S)$ is the cost of the optimal k -anonymous solution for S . Equation (1) holds because (without loss of generality) any k -anonymized group of rows in a database is at most $2k - 1$, so we can always partition the k -anonymized groups of a database into two multisets where their cardinalities are in the interval $[n/2 - 2k, n/2 + 2k]$.

Suppose we compute the optimal k -anonymity solution by evaluating equation (1) recursively, for all eligible multisubsets T . In the base case when $|S| \in [k, 2k - 1]$, we make all rows in S identical and return that solution.

We can simply enumerate all 2^n multisubsets of S to produce all possible T in equation (1). The time recurrence of the resulting algorithm is

$$T(n) \leq 2^{n+1} \cdot T(n/2 + 2k) + 2^n.$$

This recurrence solves to $T(n) \leq O(\log n \times 2^{\log n} 2^{n + \frac{n}{2} + \frac{n}{4} + \dots + 1}) \leq O(4^n \cdot \text{poly}(n))$ for constant k . □

Reminder of Theorem 4. Let ℓ be the number of attributes in a database, let c be the size of its alphabet, and let n be the number of rows. Then k -Anonymity can be solved in $2^{O(k^2(2c)^\ell)} + O(n\ell)$ time.

If ℓ and c are constants then there are at most c^ℓ possible rows. To specify a group of k -anonymized rows we write $G = \langle r', t \rangle$ where t is the number of times the anonymized row r' occurs in the group. We can think of a k -anonymous solution as a partition of the rows into such anonymized groups. The following lemma will be useful for our algorithm.

Lemma 2. Suppose that our database D contained at least $k(2k-1) \times 2^\ell$ copies of a row r . Then the optimal k -anonymity solution must contain a group containing just row r , ie. $G = \langle r, t \rangle$ where $t \geq k$.

Proof. Suppose for contradiction that our database contains more than $k(2k-1) \times 2^\ell$ copies of row r , but that our optimal solution did not contain a group $G = \langle r, t \rangle$. Without loss of generality we can assume $k \leq t \leq 2k-1$ for each group since larger groups could be divided into two groups without increasing the cost. Therefore, we must have at least $k \times 2^\ell$ groups $G = \langle r', t \rangle$ which contain the row r . Notice that each attribute of r' either matches r or is a \star . Hence, there are at most 2^ℓ possible values of r' . By the pigeonhole principle there must be at least k groups $G_i = \langle r', t_i \rangle$ containing r whose anonymized rows r' are all identical. Merge these groups into one big group $G = \langle r', \sum_{i=1}^k t_i \rangle$ at no extra cost. Each of the original k groups contained at least one copy of the row r so we can split G into two groups: $\langle r, k \rangle$ and $G' = \langle r', \sum_{i=1}^k t_i - k \rangle$ while saving at least k stars. Hence, our original solution was not optimal. Contradiction! \square

For each row r we can define $Index(r)$ to be a unique index between 0 and $c^\ell - 1$ by interpreting r as a ℓ digit number base c . Using Lemma 2, the following algorithm can be used to obtain a *kernelization* of a instance of k -Anonymity, in the sense of parameterized complexity [14].

Algorithm 1 k -anonymize a database D with small alphabet and few attributes

Require: $rowCount[i] = \|\{r \in D | Index(r) = i\}\|$

Require: c, ℓ small

$T \leftarrow k(2k) \times 2^\ell$

for Row $r \in D$ **do**

$i \leftarrow Index(r)$

if $rowCount[i] > T$ **then**

print " $\langle r, k \rangle$ " {By Lemma 2}

$rowCount[i] \leftarrow rowCount[i] - k$

end if

end for

{Now $\forall i, rowCount[i] \leq T$ so there are at most $m \leq k(2k-1)2^\ell(c^\ell) = k(2k-1)(2c)^\ell$ rows remaining}

Lemma 3. Algorithm 1 runs in $O(n\ell)$ time on a database D and outputs a database D' with at most $O(k^2(2c)^\ell)$ rows, with the property that an optimal k -anonymization for D' can be extended to an optimal k -anonymization for D in $O(n\ell)$ time.

That is, for the parameter $k + c + \ell$, the k -anonymity problem is not only fixed parameter tractable, but can also be efficiently kernelized.

Proof. (Sketch) By implementing $rowCount$ as a hash table, each $Index(r)$ and lookup operation takes $O(\ell)$ time. Hence, set up takes $O(n\ell)$ time as does the loop. By lemma 2 there must be an optimal k -anonymity solution containing $\langle r, t \rangle$ with $t \geq k$ whenever r occurs at least $k(2k-1)2^\ell$ times in D' . Therefore, if r occurs more than $k(2k)2^\ell > k + k(2k-1)2^\ell$ times in D then there is an optimal k -anonymity solution which contains the groups $\langle r, k \rangle$ and $\langle r, t \rangle$ so adding back k copies of row r to D' does not change the optimal k -anonymization except for the extra $\langle r, k \rangle$ group. \square

Proof of Theorem 4. By lemma 3, Algorithm 1 takes an arbitrary k -anonymity instance D and reduces it to a new instance D' with at most $k(2k)(2c)^\ell$ rows in time $O(n)$. We can then apply Theorem 3 to k -anonymize D' in time $O(4^m \text{poly}(m))$. The total running time is $2^{O(k^2(2c)^\ell)} + O(n\ell)$. \square

5 Hardness of 3-Anonymity With Binary Attributes

In 2005, Aggarwal *et al.* [7] showed that 3-anonymity with a ternary alphabet is NP-hard. Their proof of hardness gives a reduction from EDGE PARTITION INTO TRIANGLES, in which one is given a graph and is asked to determine if the edge set E can be partitioned into 3-sets such that each set corresponds to a copy of K_3 . In particular, Aggarwal *et al.* first present a reduction from the problem of EDGE PARTITION INTO TRIANGLES AND 4-STARS⁵ into BINARY 3-ANONYMITY. Then they introduce a third alphabet symbol to distinguish 4-stars from triangles in the reduction, concluding that a TERNARY 3-ANONYMITY algorithm can be used to solve EDGE PARTITION INTO TRIANGLES.

We shall strengthen this result by directly proving that the EDGE PARTITION INTO TRIANGLES AND 4-STARS problem is NP-Complete. In fact, we establish the hardness of edge partitioning into 4-stars on *triangle-free* graphs. Using the aforementioned reduction of Aggarwal *et al.*, the hardness of BINARY 3-ANONYMITY follows from this result.

Reminder of Theorem 5. EDGE PARTITION INTO 4-STARS is NP-Complete, even for triangle-free graphs.

We describe the setup for Theorem 5 in the following paragraphs. The reduction will be from 1-IN-3 SAT, which is well-known to be NP-Complete [15]. Recall that in the 1-IN-3 SAT problem, we are given a 3-CNF formula ϕ and are asked if there is a satisfying assignment to ϕ with the property that *exactly* one literal in each clause is true. We call a yes-instance of the problem *1-in-3 satisfiable*. Given a formula ϕ , the idea of our reduction is to create triangle-free graph gadgets— a gadget for each variable, and another type of gadget for each clause— and connect them in a (triangle-free) way such that ϕ is 1-in-3 satisfiable if and only if the resulting graph can be edge-partitioned into 4-stars. We first define a type of graph that shall be used to simulate the truth assignment of a variable in ϕ .

Definition 11. Let $d \in \mathbb{N}$ be given. The graph G_d is formed by taking two 3-Binary trees of depth d , deleting a leaf from exactly three different parents in each tree, and adding three edges so that the parents of deleted leaves in one tree are matched with the parents of deleted leaves in the other tree.

In a copy of G_d , we consider all edges adjacent to leaves to be *shared edges*, while all other edges are considered *private*. Intuitively, the shared edges are those that are *shared* with other gadgets in our final graph. We say that G contains a *share-respecting copy* of G_d if its vertex set can be partitioned into two sets S and T such that S is an induced copy of G_d , and all edges crossing the cut (S, T) are adjacent to shared edges in S .

To distinguish between the two trees in a copy of G_d , they are arbitrarily designated as the *top tree* and *bottom tree*, respectively.

The key property of the gadget G_d is given by the following claim, which says that (in a certain sense) the edges of G_d can be partitioned into 4-stars in precisely two ways. Figure 1(a) illustrates S_5 , where the dashed edges are shared and the solid edge is private. It can be found in Appendix B.

Lemma 4. Let G be a graph containing a share-respecting copy of G_d . Assuming there is an edge partition of G into 4-stars, exactly one of two cases must hold for that partition:

1. All shared edges belonging to the top tree of G_d are contained in 4-stars with centers in G_d , while all shared edges belonging to the bottom tree are contained in 4-stars with centers not contained in G_d .
2. All shared edges belonging to the bottom tree of G_d are contained in 4-stars with centers in G_d , while all shared edges belonging to the top tree are contained in 4-stars with centers not contained in G_d .

⁵ This problem is: Given a graph $G = (V, E)$, is it possible to partition the edge set E into 3-sets such that each 3-set corresponds to either a copy of K_3 or a 4-star?

In the first case of the claim, we say that the copy of G_d is *true partitioned*, and in the second case we say that G_d is *false partitioned*. Intuitively, each copy of G_d in our final graph will correspond to a variable in ϕ , and a *true/false* partition shall correspond to assigning that variable *true/false*. Lemma 4 is proved in Appendix D.

We now define another type of graph that shall be used as gadgets to represent clauses in a given 1-in-3 SAT formula.

Definition 12. *The graph S_5 is a 5-star with one of its edges labeled private and the other three edges labeled shared.*

Figure 1(b) illustrates S_5 , where the dashed edges are shared and the solid edge is private. It can be found in Appendix B.

Suppose a graph G contains a share-respecting copy of S_5 ⁶, so that one node adjacent to the private edge of S_5 has degree one. Call this node v and its adjacent node u (the center of S_5). Then, any partition of G into 4-stars must contain a 4-star with u as its center, using the edge (u, v) . But this 4-star must use two of the shared edges in S_5 . Therefore an edge-partition of G into 4-stars is possible if and only if exactly one of the shared edges in S_5 participates in a 4-star with a center that is outside of S_5 .

We are finally ready to prove Theorem 5.

Proof of Theorem 5. Let an 1-IN-3 SAT instance ϕ be given with clauses C_1, \dots, C_m and variables x_1, \dots, x_n . We wish to create a triangle-free graph G_ϕ that can be edge-partitioned into 4-stars if and only if ϕ is 1-in-3 satisfiable.

G_ϕ is constructed as follows:

- For each variable x_i , let k_i denote the number of clauses that x_i occurs in (or the number of clauses that \bar{x}_i occurs in, whichever is greater). Let d_i be the integer satisfying $3 \cdot 2^{d_i-2} < 3(k_i + 1) \leq 3 \cdot 2^{d_i-1}$. Add a copy of the graph G_{d_i} to G_ϕ , calling it A_i . Note that A_i has at least $3(k_i + 1)$ leaves.
- For each clause $C_i = (l_1 \vee l_2 \vee l_3)$, add three copies of S_5 to G_ϕ , calling them $B_{i,1}, B_{i,2}, B_{i,3}$.

Join the shared edges of these subgraphs as follows: if the literal $l_j = x_k$ is in C_i , then merge one shared edge from each of $B_{i,1}, B_{i,2}, B_{i,3}$ with three shared edges from the top tree of A_i ; otherwise, if $l_j = \bar{x}_k$ is in C_i , then merge a shared edge from each of $B_{i,1}, B_{i,2}, B_{i,3}$ with three shared edges from the bottom tree of A_i . As a heuristic use a shared edges which is incident to another unused shared edge in G_{d_k} , whenever possible.

Since A_i has a $3 \cdot 2^{d_i-1} \geq 3(k_i + 1)$ leaves, there may remain some shared edges in some A_i that have not been merged with shared edges from copy of S_5 . We deal with these extra shared edges as follows: Take three shared edges from different parents in the top tree of A_i , and merge their end vertices with a new vertex to form a 4-star. Repeat until all unused shared edges from the top are used and do the same for the shared edges on the bottom. Note that this is possible because we used three copies of S_5 for each clause; hence, shared edges from the top/bottom of each gadget are taken in multiples of three, and the number of leaves in every A_i is a multiple of three. By the above heuristic for choosing unused shared edges we will never create a multi edge.

Clearly the above reduction can be done in polynomial time. Also note that by construction, G_ϕ contains no triangles. We now argue that the formula ϕ is 1-in-3 satisfiable if and only if G_ϕ can be edge-partitioned into 4-stars. Supposing that ϕ is satisfiable, partition each variable gadget according to its assignment in a given satisfying assignment. In particular, if a variable is set to true, then *true-partition* the edges in its corresponding variable gadget. Each clause gadget can be partitioned into a 4-star, since exactly one of its shared edges are used. The remaining edges are already part of 4-stars by construction and can hence be partitioned.

For the other direction, suppose that G_ϕ can be partitioned into 4-stars. By Claim 4, each copy of G_d is either true or false partitioned. Now each clause gadget can be partitioned if and only if exactly one of its

⁶ A copy of S_5 is *share respecting* if and only if the center vertex has degree 4 and the leaf incident to the private edge has degree 1.

shared edges is used by a 4-star with a center in a variable gadget. By construction, this happens iff exactly one of the literals in the clause was assigned true in the partition for its variable gadget. Thus the partition defines a satisfying assignment for ϕ .

Finally, note that the G_ϕ constructed in Theorem 5 is triangle-free; in particular, G_ϕ is bipartite. To see this, note that each A_i is bipartite, each $B_{i,j}$ is bipartite, and for each of these subgraphs, its set of shared edges come from only one side of its bipartition. \square

While we have given a complete description above, the construction of G_ϕ is perhaps better understood through examples. We have provided two examples in Appendix B.

Corollary 1. EDGE PARTITION INTO TRIANGLES AND 4-STARS is NP-Complete.

Corollary 2. BINARY 3-ANONYMITY is NP-Complete.

Proof. Aggarwal *et al.* [7] showed that there is a polynomial time reduction from EDGE PARTITION INTO TRIANGLES AND 4-STARS to BINARY 3-ANONYMITY. Their reduction is repeated in Appendix C for completeness. \square

6 Hardness of Computing ℓ -diversity

Finally, we consider an alternative privacy model called ℓ -diversity, which strengthens the privacy guarantees of the k -anonymity model. It was first proposed to prevent certain background knowledge attacks which could potentially be used against a k -anonymized dataset [5]. In the model, we distinguish between which attributes of the database are merely potentially identifying and which are highly sensitive. Those which are highly sensitive require a strong privacy guarantee.

Definition 13. The cost of a ℓ -diverse solutions is the number of stars introduced, among the attributes $q \in Q$, to the database.

The fact that optimal 3-diversity with binary attributes and one sensitive ternary attribute is NP-hard should not be too surprising, in light of our proofs of hardness for 3-anonymity. Intuitively, the extra sensitive attribute constraint should make 3-diversity only harder than 3-anonymity. What is perhaps surprising is that optimal 2-diversity is NP-hard for databases with three sensitive attributes per row, in light of our result that optimal 2-anonymity is in P .

Reminder of Theorem 6. Optimal 2-diversity with binary attributes and three sensitive attributes is NP-hard.

Proof. The reduction is from edge partition into triangles which is known to be NP-Complete even when the graph is tripartite [16]. The idea for the reduction is similar to the reductions in [7] for binary k -anonymity (see Appendix 6). Given a graph $G = (V, E)$, define a 2-diversity instance as follows: the rows of the table correspond to each $e \in E$, while the columns correspond to the $n = |V|$ vertices of G plus the sensitive attributes $s_{i_0}, s_{i_1}, s_{i_2}$. Given an arbitrary ordering of the vertices $V = \{v_1, \dots, v_n\}$ and edges $E = \{e_1, \dots, e_m\}$ define a matrix R^G as follows:

$$R^G[i][j] = \begin{cases} 1 & \text{if } v_j \in e_i; \\ 0 & \text{otherwise.} \end{cases}$$

Let V_0, V_1, V_2 be the tripartition of vertices in G . Now label:

$$s_{i_j} = \begin{cases} 0 & \text{if } e_i \cap V_j = \emptyset; \\ 1 & \text{otherwise;} \end{cases}$$

The cost of grouping any three rows in a 2-diverse solution is at least three stars because the graph is simple. Furthermore, any group of more than three rows will require more than three stars per row to 2-diversify. The proof is argument is identical to lemma 6 in Appendix C.

Lemma 5. *Any group of only two rows in R^G violates the 2-diversity constraint.*

Proof. Let i, j be any pair of distinct rows. Because the graph is tripartite, either $s_{i_1} = s_{j_1}$ or $s_{i_2} = s_{j_2}$ or else $s_{i_3} = s_{j_3}$. The diversity constraints for two rows will look like:

	Q	S
e_i	...	110
e_j	...	101

□

Similarly, the diversity constraints coupled with the fact that the graph is 3-Partite also prevent us from choosing three rows corresponding to a 4-star in G because the rows would share a sensitive attribute. However, the diversity constraints do allow for the possibility that the three rows correspond to a triangle in G as illustrated in the table:

	Q	S
e_i	1	10
e_j	1	01
e_k	0	11

Thus the edges of G can be partitioned into triangles iff the 2-diversity instance has a solution that introduces exactly 3 stars per row. □

Reminder of Theorem 7. *Optimal 3-diversity with binary attributes is NP hard, with only one sensitive ternary attribute.*

Proof. The hardness reduction for 3-diversity with one sensitive attribute is essentially the same as above. Assume that G is tripartite, and let V_1, V_2, V_3 be the three partite sets in G . Let s_i denote the sensitive attribute for row v_i . If 3-diversity is to be feasible then the sensitive attribute s_i must be allowed to take at least three values. Other attributes must be binary.

$$s_i = \begin{cases} 1 & \text{if } e_i = (x, y), \text{ with } x \in V_1, y \in V_2; \\ 2 & \text{if } e_i = (x, z), \text{ with } x \in V_1, z \in V_3; \\ 3 & \text{if } e_i = (y, z), \text{ with } y \in V_2, z \in V_3; \end{cases}$$

As before, the diversity constraints now prevent us from grouping three rows which correspond to a 4-star. Groups of rows which do not correspond to a triangle in G still require more than three stars per row. Thus the edges of G can be partitioned into triangles iff the 3-diversity instance has a solution that introduces exactly 3 stars per row. □

7 Conclusion

We have demonstrated the hardness and feasibility of several methods used in database privacy, settling several open problems on the topic. The upshot is that most of these problems are difficult to solve optimally, even in very special cases; however in some interesting cases these problems can be solved faster. Several interesting open questions address possible ways around this intractability:

- To what degree can the hard problems be approximately solved? For example, the best known approximation algorithm for k -anonymity, given by Park and Shim [17], suppresses no more than $O(\log k)$ times the optimal number of entries. Could better approximation ratios be achieved when the number of attributes is small?

- The best known running time for Simplex Matching is $O(n^3 + n^2m^2)$ steps [9]. Here, n is the number of nodes and m is the number of hyperedges in the hypergraph. In our algorithm for 2-anonymity, n is also the number of rows in the database while $m = \binom{n}{3} = O(n^3)$ because we add a hyperedge for every triples. Hence our algorithm for 2-Anonymity has running time $O(n^8)$. Can this exponent be reduced to a more practical running time?

Acknowledgments

We would like to thank Manuel Blum, Lenore Blum and Anupam Gupta for their help and guidance during this work.

References

1. Agrawal, R., Srikant, R.: Privacy-preserving data mining. *ACM SIGMOD Rec.* **29**(2) (2000) 439–450
2. Dwork, C.: Differential privacy. *International Colloquium on Automata, Languages and Programming (ICALP)* (2006) 1–12
3. Samarati, P.: Protecting Respondents’ Identities in Microdata Release. *IEEE Transactions on Knowledge and Data Engineering* (2001) 1010–1027
4. Sweeney, L.: k-anonymity: A model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems* **10**(5) (2002) 557–570
5. Machanavajjhala, A., Kifer, D., Gehrke, J., Venkitasubramaniam, M.: l-diversity: Privacy beyond k-anonymity. (2007)
6. Meyerson, A., Williams, R.: On the complexity of optimal K-anonymity. *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems* (2004) 223–228
7. Aggarwal, G., Feder, T., Kenthapadi, K., Motwani, R., Panigrahy, R., Thomas, D., Zhu, A.: Anonymizing tables. *Proceedings of the 10th International Conference on Database Theory* (2005) 246–258
8. Bonizzoni, P., Della Vedova, G., Dondi, R.: The k-Anonymity Problem is Hard. *Proceedings of 17th International Symposium on Fundamentals of Computation Theory* (2009) 26–37
9. Anshelevich, E., Karagiozova, A.: Terminal backup, 3D matching, and covering cubic graphs. *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing* (2007) 391–400
10. Chaytor, R., Evans, P., Wareham, T.: Fixed-Parameter Tractability of Anonymizing Data by Suppressing Entries. *Lecture Notes in Computer Science* **5165** (2008) 23–31
11. Dor, D., Tarsi, M.: Graph decomposition is NPC - A complete proof of Holyer’s conjecture. In: *Proceedings of the twenty fourth annual ACM symposium on Theory of computing*, ACM (1992) 252–263
12. Papadimitriou, C., Yannakakis, M.: Optimization, approximation, and complexity classes. In: *Proceedings of the twentieth annual ACM symposium on Theory of computing*, ACM (1988) 234
13. Kann, V.: Maximum Bounded 3-Dimensional Matching in MAX SNP-Complete. *Information Processing Letters* **37**(1) (1991) 27–35
14. Flum, J., Grohe, M.: *Parameterized complexity theory*. Springer-Verlag New York Inc (2006)
15. Schaefer, T.: The complexity of satisfiability problems. *Proceedings of the tenth annual ACM symposium on Theory of computing* (1978) 216–226
16. Garey, M., Johnson, D.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. WH Freeman & Co. New York, NY, USA (1979)
17. Park, H., Shim, K.: Approximate algorithms for K-anonymity. *Proceedings of the 2007 ACM SIGMOD international conference on Management of data* (2007) 67–78

A Proof of Lemma 1

Recall we had the following three properties of the database D in the reduction of Theorem 2:

1. For any x rows r_i, r_j, r_k, r_l , where $x \geq 4$, the cost of anonymizing these rows is

$$C_{i,j,k,l} = x \cdot 27$$

because there is no alphabet symbol that is used in all 4 rows.

2. If $\{r_i, r_j, r_k\} \notin M$ then the cost of anonymizing the three corresponding rows is

$$C_{i,j,k} = 3 \cdot 27$$

because there is no alphabet symbol that is used in all 3 rows.

3. If $\{r_i, r_j, r_k\} \in M$ then the cost of anonymizing the three corresponding rows is

$$C_{i,j,k} = 3 \cdot 26$$

because the three rows will match in exactly one of the 27 columns.

Reminder of Lemma 1. *There is a polynomial time mapping g from 3DM-3 feasible solutions to 3-anonymity feasible solutions, such that if $M' \subseteq M$ is a 3DM-3 feasible solution then $C_{3DM}(M') = 27C_{3ANON}(g(M'))$.*

Proof. We use the reduction defined in the proof of Theorem 2. Recall that in a 3-anonymity solution P is a partition of the rows into groups of size 3, 4 and 5. By the three properties of D , any group which does not correspond to triple from M must be suppressed entirely. Hence, we can think of the solution as a partition of the rows into triples (x_i, y_j, w_k) from M and some other rows. Similarly, we can think of a 3DM solution as a partition of the elements into triples from M and some other elements. Thus we can define a polynomial time computable transformation f between 3DM-3 solutions and 3-anonymity solutions.

By the above properties of D , $Cost_3^*(g(M')) = 27n - 3|M'|$. Therefore,

$$\begin{aligned} C_{3DM}(M') &= \frac{3|M'|}{|X| + |Y| + |W|} \\ &= \frac{3|M'|}{n} \\ &= 27 - \frac{27n - 3|M'|}{n} \\ &= 27 - \frac{Cost_3^*(g(M'))}{n} \\ &= 27 \cdot C_{3ANON}(g(M')) \end{aligned}$$

□

B Edge Partition into 4-Star Reduction - Examples

Figure 1 shows examples of a variable gadget and a clause gadget.

Example 1. $\phi = (\bar{x}, y, z)(x, \bar{y}, z)$. Note that the 1-IN-3 SAT formula has two satisfying assignments: $(x = t, y = t, z = f)$, $(x = f, y = f, z = f)$. Similarly, the corresponding graph G_ϕ (shown in figure 2) can be partitioned into 4-stars in exactly two ways, both corresponding to the satisfying assignments.

Example 2. $\phi = (x, y, z)(\bar{x}, \bar{y}, \bar{z})$. Note that ϕ is not 1-in-3 satisfiable. Similarly, the corresponding graph G_ϕ (shown in figure 3) cannot be edge-partitioned into 4-stars.

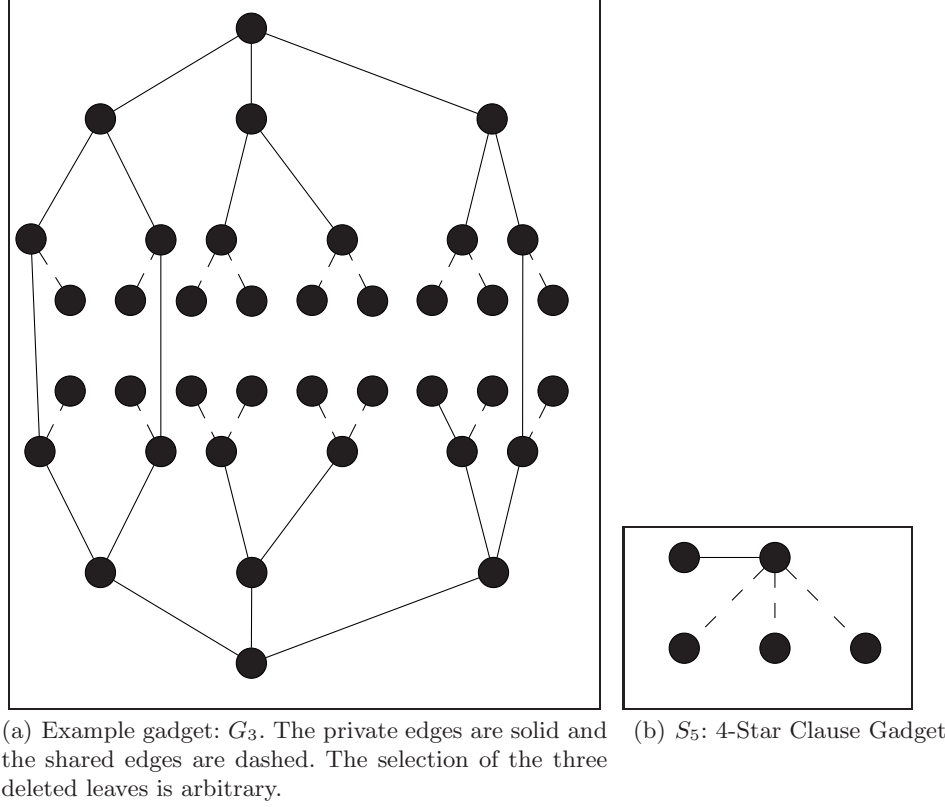


Fig. 1. Gadgets

C Reducing Edge Partition Into Triangles And 4-Stars to Binary 3-Anonymity

Given a graph $G = (V, E)$ with m edges and n vertices build the following table: the rows of the table correspond to each edge $e \in E$, while the columns correspond to the $n = |V|$ vertices of G . Given an arbitrary ordering of the vertices $V = \{v_1, \dots, v_n\}$ and edges $E = \{e_1, \dots, e_m\}$ define a database R^G as follows:

$$R^G[i][j] = \begin{cases} 1 & \text{if } v_j \in e_i; \\ 0 & \text{otherwise.} \end{cases}$$

Clearly this reduction takes polynomial time. Note that, because the graph G is simple, any 3-anonymous solution must include at least three stars per row. This follows because for any set of three edges, there are at least three vertices that are incident with one, but not all, of the three edges. Furthermore, note that if a set of three edges do not form a triangle or 4-star, then there are at least four vertices that are incident with one (but not all) of the three edges. The result follows from lemma 6.

Lemma 6. *Let m be the number of edges in G , the cost of the optimal 3-anonymous solution for R^G is $3m$ stars iff the graph G can be edge partitioned into 4-stars and triangles.*

Proof. First, suppose that the cost of the optimal 3-ANONYMITY solution is $3m$. Since each row has at least 3 stars in it, each row must have exactly three stars because there are m rows in R^G . Given a set of three identical (anonymized) rows, each row has 3 stars each corresponding to a vertex that was incident to one, but not all of the three edges represented by those rows. Hence, those three edges form either a 4-star or a triangle. Therefore, the edges of the graph can be partitioned into triangles and 4-stars.

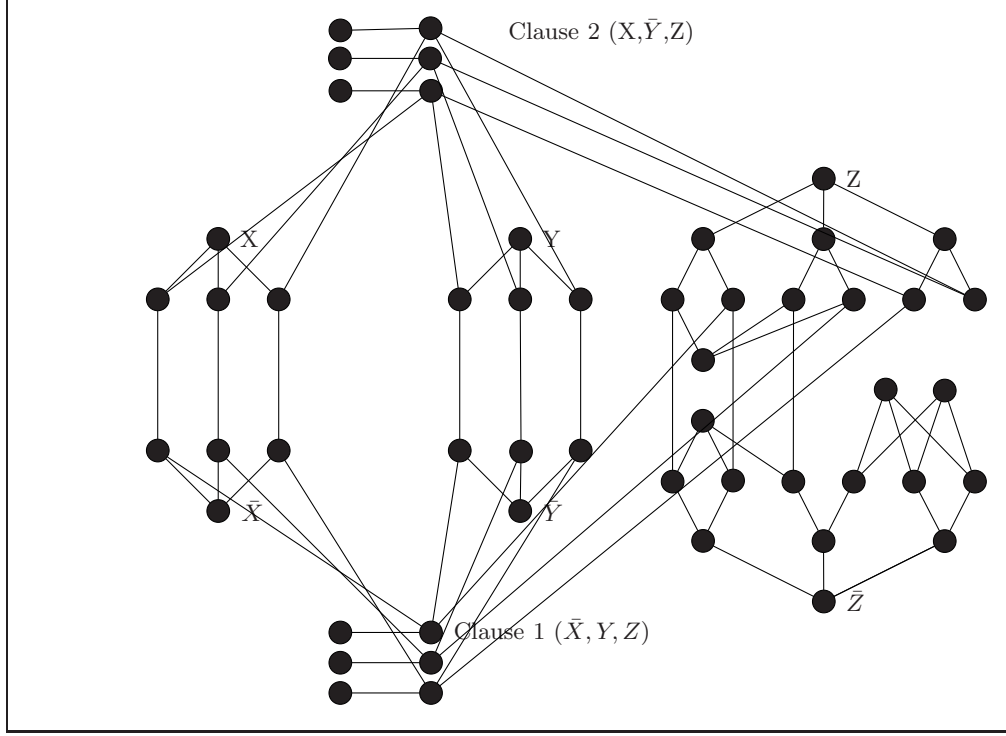


Fig. 2. $\phi = (\bar{x}, y, z)(x, \bar{y}, z)$

For the other direction, suppose that G can be partitioned into triangles and 4-stars. Group the rows of the 3-anonymity instance according to the partition. Now consider a group of three rows in the table that correspond to the edges of a 4-star. The three rows have the form:

(v_0, v_1)	$\cdots 1100 \cdots$
(v_0, v_2)	$\cdots 1010 \cdots$
(v_0, v_3)	$\cdots 1001 \cdots$

where the \cdots are all 0's. For a triangle, the three rows corresponding to its edges looks like:

(v_0, v_1)	$\cdots 110 \cdots$
(v_0, v_2)	$\cdots 101 \cdots$
(v_1, v_2)	$\cdots 011 \cdots$

where again the \cdots are all 0's. Clearly, both groups of rows can be made identical by suppressing only three entries per row. Hence, the table can be made 3-anonymous with $3m$ stars. \square

D Edge Partitioning G_d into 4-Stars

Recall the statement of lemma 4

Lemma 7. *Let G be a graph containing a share-respecting copy of G_d . Assuming there is an edge partition of G into 4-stars, exactly one of two cases must hold for that partition:*

1. *All shared edges belonging to the top tree of G_d are contained in 4-stars with centers in G_d , while all shared edges belonging to the bottom tree are contained in 4-stars with centers not contained in G_d .*

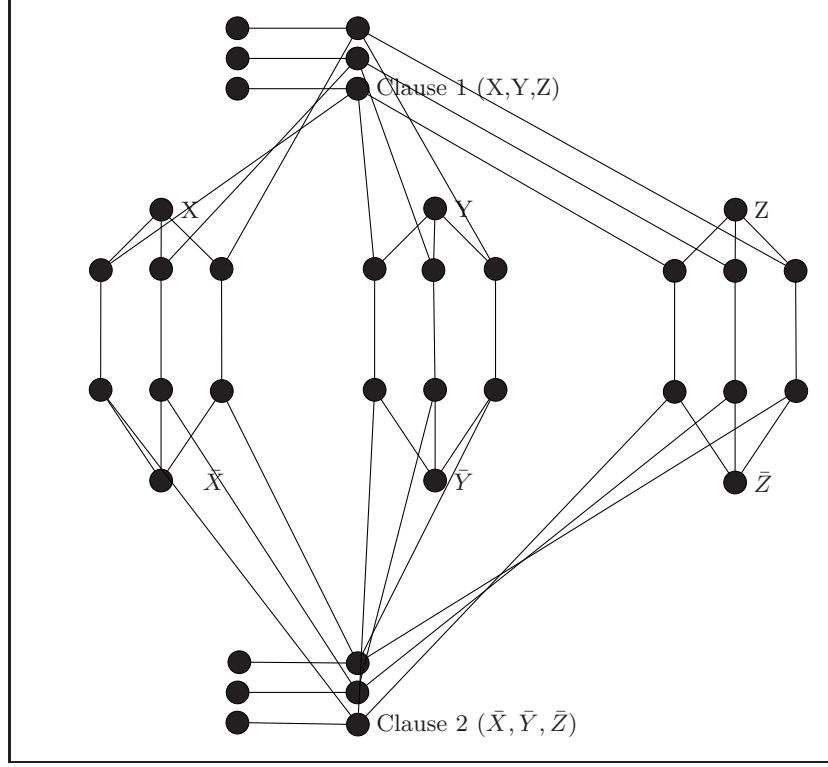


Fig. 3. $\phi = (\mathbf{x}, \mathbf{y}, \mathbf{z})(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\mathbf{z}})$

2. All shared edges belonging to the bottom tree of G_d are contained in 4-stars with centers in G_d , while all shared edges belonging to the bottom tree are contained in 4-stars with centers not contained in G_d .

Proof. Let G be a graph which contains a share-respecting copy of G_d along with an edge partition P of G into 4-stars. Note that every vertex in G_d has degree 3, even the leaves in G_d have two more shared edges in G_d . If an internal vertex x in a 3-Binary Tree is the center of a 4-star in P (see figure D) then its parent y cannot be the center of any 4-star in P because its degree has been reduced to 2. This means that z must be the center of a 4-star in P to cover the edge (y, z) . Similarly, if x was not the center of a 4-star then y must be the center of a 4-star to cover the edge (x, y) .

Now the pattern becomes evident: the parent of z cannot be the center of a 4-star so z 's grandparent must be the center of a 4-star, and so on. Therefore, in any edge partition, if there is a 4-star centered at vertex v at depth i , then the ancestors of v at depths $i - 2, i - 4, \dots$ as well as the descendants at depths $i + 2, i + 4, \dots$ must *all* be centers of 4-stars as well.

Consider the root of the 3-Binary Tree at depth 0, there are only two possible scenarios. Scenario 1, the root is the center of a 4-star and all the vertices (descendants) at depths 0, 2, 4, \dots in that 3-Binary Tree must also be the centers of 4-stars. Scenario 2, the root is not the center of a 4-star and all the vertices (descendants) at depths 1, 3, \dots must be centers of 4-stars.

By construction of G_d there must be exactly three edges between the top and bottom 3-Binary Trees in G_D . Pick one such edge (u, v) , any edge partition of G must use the edge (u, v) so either u or v must be the center of a 4-star. Without loss of generality assume that u is the center of a 4-star and that u is in the bottom 3-Binary Tree. Notice that both u and v are at depth $d - 1$ in their respective trees. Assume that d is odd (the proof is similar for d even), then in the bottom tree we are in Scenario 1, but in the top tree we are in Scenario 2. All shared edges belonging to the bottom tree of G_d are contained in 4-stars centered

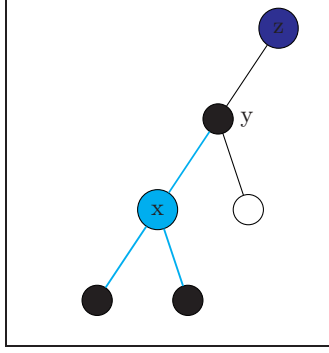


Fig. 4. x is the center of a 4-star, therefore y cannot be the center of a 4-star. In any partition the edge (y, z) must be covered by a 4-star centered at z .

at depth $d - 1$, while no shared edges from the top tree of G_d can be contained in 4-stars centered at depth $d - 1$. \square